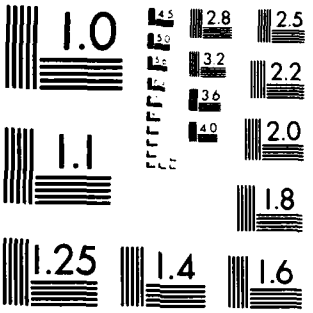
AD-A142 306 ELLIPTIC SOLVERS FOR MEDITERRANEAN SEA OCEAN MODELING 1/
(U) JAYCOR ALEXANDRIA VA INFORMATION SYSTEMS DIV
A J WALLCRAFT MAY 84 N00014-83-C-0681

UNCLASSIFIED F/G 8/3 NL

END
DATE FILMED
7-84
DTIC



(12)

ELLIPTIC SOLVERS FOR
MEDITERRANEAN SEA OCEAN MODELING

JAYCOR

205 South Whiting Street
Alexandria, Virginia 22304

AD-A142 306

ELLIPTIC SOLVERS FOR MEDITERRANEAN SEA OCEAN MODELING

Prepared by

Alan J. Wallcraft

JAYCOR

May 1984

Prepared for:

Office of Naval Research

Under:

Contract Number N00014-83-C-0681

ELLIPTIC SOLVERS FOR MEDITERRANEAN SEA OCEAN MODELING

Alan J. Wallcraft

A-1

TABLE OF CONTENTS

I. MEDITERRANEAN BASIN SOLVERS

Versions of two beta-plane ocean models suitable for use in the Mediterranean have been prepared. One has two-layers and topography, and the other is a reduced gravity one active layer model. They use both the direct and the preconditioned capacitance matrix techniques (DCMT and PCMT) for solving the constituent non-rectangular two dimensional finite difference Helmholtz equations with Neumann boundary conditions. Because the ocean models use the primitive equations there are no special problems associated with the modeling of regions with islands, as there would be, for example, in a model using the vorticity stream-function formulation. DCMT is generally faster but requires more memory than the preconditioned method, but PCMT is particularly suited to problems with large Helmholtz coefficients. Therefore the ocean models have been set up to use DCMT only for the external gravity wave mode, all other modes being solved by the PCMT to save storage (these modes have a very large Helmholtz coefficient). Appendix 1 contains a listing of the subroutine that interfaces the ocean models and the Helmholtz solvers. This is the only solver subroutine that need be modified by an ocean modeler when implementing a new model geometry, and the modifications (fully documented within the routine) only consist of redefining a few PARAMETER values. The full ocean model and solver code listings are far too long to include in this report, but the codes have been delivered to NORDA. Figure 1 shows a result of a test case Western Mediterranean ocean model simulation. The grid resolution used in this experiment was very coarse, the simulation was designed to test the model code rather than actually realistically represent the circulation in the Mediterranean.

II. SOLVER FOR FLOW OVER SILLS

NORDA's existing 2-D Boussinesq flow models are for a flat bottom only, they require the solution of Helmholtz's equation each timestep to obtain the stream-function. A very efficient elliptic solver for this model had previously been developed by JAYCOR, it used the FACR(0) method and solved Helmholtz's equation over a rectangular region with Dirichlet boundary conditions on the "bottom" and "top" boundaries, and periodic conditions on the other boundaries. This solver was extended, using the direct capacitance matrix technique, to allow a simple triangular "mountain" to rise above the otherwise flat bottom. The solver has been fully tested and has been added to the 2-D Boussinesq flow model.

III. SOLVERS ON THE FPS-164

NORDA has ordered an FPS-164 but it has not been installed yet, this work was therefore performed at the NATO SACLANT center in Italy. The FPS-164 is an array processor that is attached to a host device, in this case a VAX 750. As is was set up at SACLANT, it was run in Single Job Executive mode (SJE), which meant that it was regarded as a batch machine running one program at a time that was submitted from the VAX. A

useful feature of this arrangement was that compilation and link editing of the program took place on the host machine by means of a cross compiler and linker. This had the effect of freeing the FPS-164 to run only load modules (called images) that had been successfully compiled and linked. Thus errors in compilation and linkage could be detected at the VAX interactive use level and corrected without requiring access to the array processor. In addition this meant that the source program, linkage instructions and FPS-164 execution modules could be kept on the VAX file base and edited using the VAX editors. This greatly speeded up program development. A slight drawback when using the FPS-164 to evaluate the speed at which code would run was the absence of an explicit run time clock communicating with the VAX operating system. This prevented an explicit time limit on execution being set externally to the program and stopped timings being deduced from run time figures as these were not available. There is an internal cycle counter that can be reset and read by system subroutine calls (CALL SYS$CLEAR, CALL SYS$READ, etc.) but these had to be invoked explicitly to get timing information. The compiler had 5 levels of optimization that could be invoked, levels 0 to 4. Only the two highest levels exploited the array processor architecture, the highest ignoring potential storage clashes and generally not recommended for use except with extreme caution. In practice, code development proceeded by getting test results first on the VAX, then from the FPS-164 with optimization set to 0 or 1, then when these agreed, getting results for optimization 3 and then 4.

The solvers detailed in I and II above were moved onto the FPS-164, in general they ran about 10 times faster on the FPS than on the VAX 11/750 (without a floating point accelerator board), or about 10 times slower than on the TIASC supercomputer.

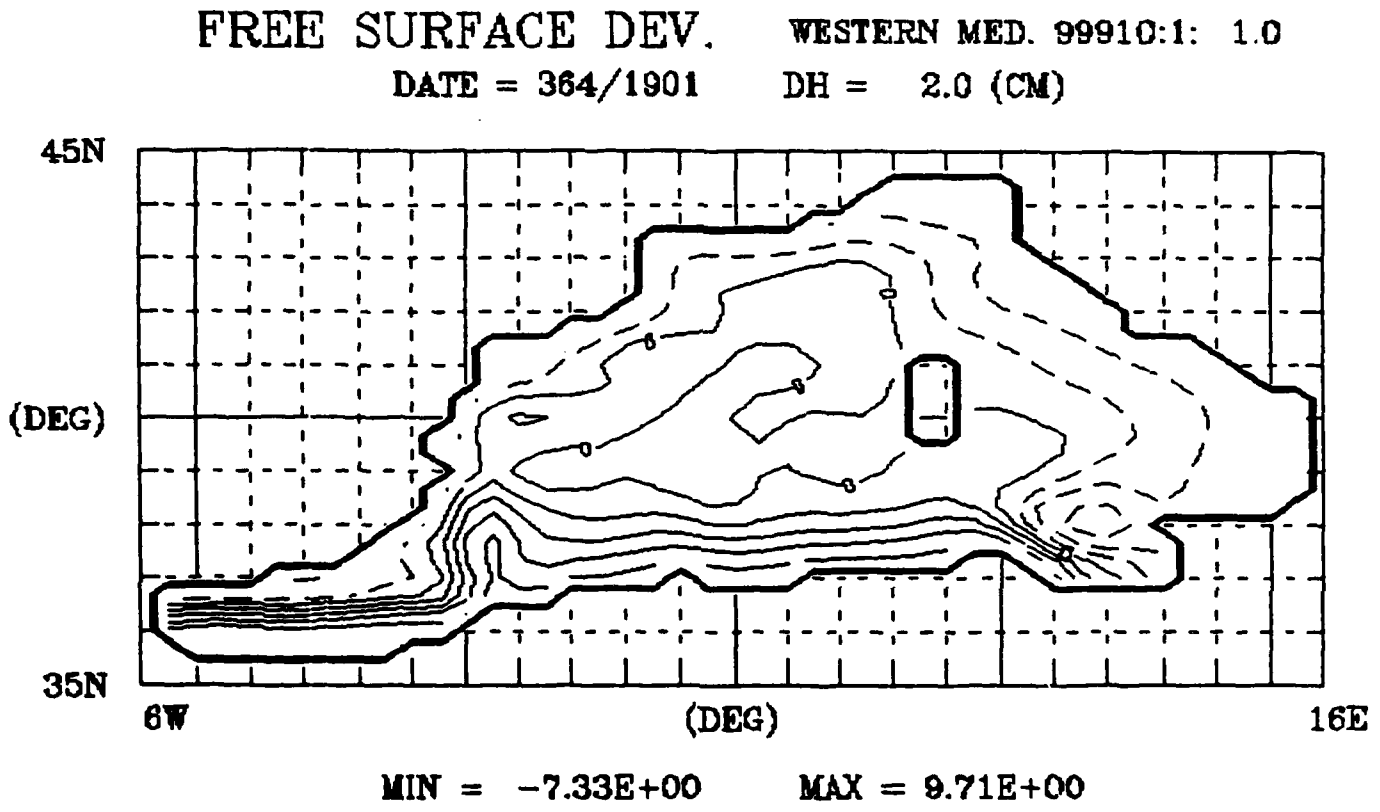


Figure 1. Example instantaneous free surface deviation from a coarse grid Western Mediterranean ocean model. The experiment was designed to test the model code rather than to realistically simulate the flow in the Mediterranean.

APPENDIX 1

LISTING OF THE ROUTINE THAT INTERFACES THE OCEAN MODELS TO THE HELMHOLTZ SOLVERS

```
      SUBROUTINE CHSLVX(HD,HT,RS, AX,AY,AC,ACKL, KK,LL)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (IH=045, JH=029,  KH=002)
      PARAMETER (IU=IH-1,JV=JH-1,
     +           IV=IH-2,JU=JH-2)
C
      PARAMETER (JHS=34,JOS=1, JOSM=JOS-1)
      PARAMETER (MXKP=112)
      PARAMETER (MXKC1=112*113,MXKC2=21*112, MXKC3=3*112,
     +           KWSP=21*(112+2*21+6)
C
      PARAMETER (NX=IH-2, NY=JHS-2, KQ=NY*((NX+7)/4+1)+(NY+3)/2+8)
C
      DOUBLE PRECISION AX,AY,AC(KH),ACKL
      DIMENSION HD(IH,JH),HT(IH,JH),RS(IH,JH)
C
      COMMON/BV/ W1(IH,JHS),W2(IH,JHS),W3(IH,JHS),W4(IH,JHS)
      DOUBLE PRECISION WQ
      DIMENSION MAP(IH,JHS),WQ(JHS,5),WC(KWSP)
      EQUIVALENCE (W1(1),MAP(1)), (W3(1),WQ(1)), (W4(1),WC(1))
C
C**********
C*
C 1)  MULTIPLE ENTRY ROUTINE FOR CARTESIAN HELMHOLTZ SOLVER(S).
C
C 2)  THIS VERSION FOR THE CAPACITANCE MATRIX TECHNIQUE SOLVERS.
C
C 3)  PARAMETERS CONTROLLING SOLVER:
C        JHS   = 2-ND DIM. OF RECTANGLE FOR SOLVER (.GE.JH).
C                 (JHS-2 MUST BE OF FORM P*2**Q WITH P=4,5, OR 6).
C        JOS   = RS(1,1) IS AT W1(1,JOS)  (JOS = 1 OR JHS-JV),
C                 CHOOSE JOS TO MINIMISE MXKP.
C        MXKP  = MAX. NO. OF IRREGULAR BOUNDARY POINTS.
C        MXKC? = MAX. SIZE OF CV:
C              MXKC = MXKP*(MXKP+1) - THE DIRECT C.M.T.
C              MXKC = MXKP*KBW      - THE PRECONDITIONED C.M.T. WITH
C                                     BANDWIDTH KBW
C        MXKC1 = MAX SIZE OF CV FOR LEAPFROG T.S. AND MODE NO. 1
C                 ( TYPICALLY MXKC1 = MXKP*(MXKP+1), OR
C                                     MXKP*(2*SQRT(MXKP)+1) ),
C        MXKC2 = MAX SIZE OF CV FOR LEAPFROG T.S. AND ALL OTHER MODES
C                 ( TYPICALLY MXKC1 .GE. MXKC2 .GE .MXKC3 ),
C        MXKC3 = MAX SIZE OF CV FOR RESTART  T.S.
C                 ( TYPICALLY MXKC3 = MXKP*3 ).
C        KWSP = FOR D.C.M.T.  6*MXKP,
C               FOR P.C.M.T.  MAX(6*MXKP,KBW*(MXKP+2*KBW+6)).
C
C        ALSO IH MUST BE ODD FOR THIS SOLVER.
C
C 4)  FOR RECTANGULAR REGIONS SET MXKP=MXKC?=1  (PROVIDED JHS=JH).
C
C 5)  COMMON AREA /BV/ IS USED FOR WORKSPACE, IT PROBABLY HAS A
C      DIFFERENT LENGTH IN MODEL DRIVER SUBROUTINES 'AA****'.
C     'RS' IS USED AS WORKSPACE WHEN CALLING 'NB*CMT', THIS MEANS:
C        A) IH*JH MUST BE .GE. 6*KP (TESTED IN 'CHSLVI'),
C        B) 'RS' IS OVERWRITTEN ON EXIT FROM 'CHSLVS'.
C
C 6)  ALAN J. WALLCRAFT,   DECEMBER 1983.
C*
```

```
C**********
C
      COMMON/SEA/ I1,INX,J1,JNY
      COMMON/CCXXXX/  QQH(KQ,KH),CV(MXKC1+MXKC2*(KH-1)+MXKC3*KH),
     +   EV(2,KH,2),CO(5*MXKP,KH,2),CW(MXKP,KH)
      COMMON/CIXXXX/  ND(MXKP),KC(KH,2),MC(KH,2),KP,KR,LLOLD
      SAVE /SEA/, /CCXXXX/, /CIXXXX/
C
      DOUBLE PRECISION AP,ACSCAL
C
      DATA RELERR / 1.E-6 /
      DATA ZERO, ONE / 0.0, 1.0 /
C
C     DUMMY HEADER ENTRY.
C
      LLOLD = 2
      RETURN
C     END OF (DUMMY) ENTRY CHSLVX.
C*****
      ENTRY CHSLVI(HD, AX,AY,AC)
C
C     INITIALISE SOLVER(S) FOR ALL LAYERS AND TIME-STEPS.
C
      IF     (JHS.LT.JH) THEN
        WRITE(6,9000) JHS,JH
        STOP
      ELSEIF (JOS.NE.1 .AND. JOS.NE.JHS-JV) THEN
        WRITE(6,9100) JOS,JHS-JV
        STOP
      ENDIF
C
      KC(1,1) = MXKC3
      MC(1,1) = 1
      IF(KH.LT.2) GOTO 113
      DO 13 K=2,KH
        KC(K,1) = MXKC3
        MC(K,1) = MC(K-1,1) + KC(K-1,1)
   13 CONTINUE
  113 CONTINUE
      KC(1,2) = MXKC1
      MC(1,2) = MC(KH,1) + KC(KH,1)
      IF(KH.LT.2) GOTO 114
      DO 14 K=2,KH
        KC(K,2) = MXKC2
        MC(K,2) = MC(K-1,2) + KC(K-1,2)
   14 CONTINUE
  114 CONTINUE
C
      DO 15 K=1,KH
      DO 15 I=1,MXKP
        CW(I,K) = ZERO
   15 CONTINUE
C
      I1  =     2
      INX = IH -1
      J1  =     2
      JNY = JHS-1
      KP  =     0
      DO 16 L=1,2
      IF     (L.EQ.1) THEN
```

```
          ACSCAL = 4.DO
        ELSE
          ACSCAL = 1.DO
        ENDIF
        DO 16 K=1,KH
          DO 11 J=1,JHS
          DO 11 I=1,IH
            MAP(I,J) = 1
   11     CONTINUE
          DO 12 J=1,JH
          DO 12 I=1,IH
            MAP(I,J+JOSM) = 1 - MAX(ZERO,MIN(HD(I,J),ONE))
   12     CONTINUE
          AP = -(AX + AX + AY + AY + ACSCAL*AC(K))
          CALL NBREGC(IH,JHS,MAP, AX,AY,AP,
     +                KR,MXKP,KP,CO(1,K,L),ND,
     +                KQ,QQH(1,K), WQ)
          IF      (KP.GT.0) THEN
            IF (KC(K,L).GE.KP*(KP+1)) THEN
              CALL NBDCAP(KP,KC(K,L),CV(MC(K,L)),          CO(1,K,L),ND,
     +                    KR,W1,KQ,QQH(1,K  ,W3,KWSP,WC)
            ELSE
              CALL NBPCAP(KP,KC(K,L),CV(MC  ,L)),EV(1,K,L),CO(1,K,L),ND,
     +                    KR,W1,KQ,QQH(1,K  W2,W3,KWSP,WC)
            ENDIF
          ENDIF
   16   CONTINUE
C
        IF      (6*KP.GT.IH*JH) THEN
          WRITE(6,9200) KP,(IH*JH)/6
          STOP
        ENDIF
C
        IF      (KP.GT.0) THEN
          CALL NBMAPS(IH,JHS,W1, KP,CO,ND, W2, 2)
C
          DO 17 L=1,2
            IF      (L.EQ.1) THEN
              WRITE(6,6000)
            ELSE
              WRITE(6,6050)
            ENDIF
            DO 17 K=1,KH
              KBW = KC(K,L)/KP
              IF      (KBW.GE.KP+1) THEN
                WRITE(6,6100) K
              ELSE
                WRITE(6,6200) K,KBW-MOD(KBW+1,2)
              ENDIF
   17     CONTINUE
        ELSE
          WRITE(6,7000)
        ENDIF
C
        LLOLD = 2
        RETURN
C       END OF ENTRY CHSLVI.
C*****
        ENTRY CHSLVS(HT,RS, KK,LL, AX,AY,ACKL)
C
```

```
C     SOLVES SINGLE HELMHOLTZ'S EQUATION .
C
      IF     (LLOLD.NE.LL) THEN
        AP = -(AX + AX + AY + AY + ACKL)
        CALL CHLMNN(IH,JHS,AX,AY,AP, KQ,QQH(1,KK),  WQ)
        IF     (KK.EQ.KH) THEN
          LLOLD = LL
        ENDIF
      ENDIF
C
      IF     (KP.LE.0) THEN
        DO 21 J=1,JH
        DO 21 I=1,IH
          HT(I,J) = RS(I,J)
   21   CONTINUE
        CALL HELMNN(KR,HT, KQ,QQH(1,KK),W3,W4)
      ELSE
        IF(JOSM.LT.1) GOTO 131
        DO 31 J=1,JOSM
        DO 31 I=1,IH
          W1(I,J) = ZERO
   31   CONTINUE
  131   CONTINUE
        IF(JHS.LT.JH+1) GOTO 132
        DO 32 J=JH+1,JHS
        DO 32 I=1,IH
          W1(I,J) = ZERO
   32   CONTINUE
  132   CONTINUE
        DO 33 J=1,JH
        DO 33 I=1,IH
          W1(I,J+JOSM) = RS(I,J)
   33   CONTINUE
C
        IF     (KC(KK,LL).GE.KP*(KP+1)) THEN
          CALL NBDCMT(KR,W2,W1,  W1,
     +                KQ,QQH(1,KK),W3,W4,
     +                KP,KC(KK,LL),CV(MC(KK,LL)),
     +                CO(1,KK,LL),ND, CW(1,KK),  RS)
        ELSE
          CALL NBPCMT(KR,W2,W1, RELERR, W1,
     +                KQ,QQH(1,KK),W3,W4,
     +                KP,KC(KK,LL),CV(MC(KK,LL)),EV(1,KK,LL),
     +                CO(1,KK,LL),ND, CW(1,KK),  RS)
        ENDIF
C
        DO 41 J=1,JH
        DO 41 I=1,IH
          HT(I,J) = W2(I,J+JOSM)
   41   CONTINUE
      ENDIF
      RETURN
 6000 FORMAT(/// 5X,'RESTART  TIMESTEP SOLVERS:' //)
 6050 FORMAT(//  5X,'LEAPFROG TIMESTEP SOLVERS:' //)
 6100 FORMAT(20X,'K =',I2,5X,'DIRECT          C.M.T. SOLVER.' /)
 6200 FORMAT(20X,'K =',I2,5X,'PRECONDITIONED C.M.T. SOLVER',
     +   '  (BANDWIDTH =',I3,').' /)
 7000 FORMAT(// 20X,'RECTANGULAR REGION (AND RECTANGULAR SOLVER).' //)
 9000 FORMAT(// 10X,'*****  ERROR IN CHSLVI  -  ',
     +   'PARAMETER JHS =',I4,'  MUST BE .GE. JH =',I4,'  *****' //)
```

```
 9100 FORMAT(// 10X,'*****  ERROR IN CHSLVI  -  ',
     +   'PARAMETER JOS =',I4,'  MUST BE 1 OR JH-JV (=',I4,
     +   ')  *****' //)
 9200 FORMAT(// 10X,'*****  ERROR IN CHSLVI  -  ',
     +   'CHSLVS ASSUMES THAT KP =',I4,'  IS .LE. (IH*JH)/6 =',I4 //
     +   50X,'(IF KP IS CORRECT RS CANNOT BE USED IN CALLS TO NB*CMT',
     +   '  -  RECODE CHSLVX)  *****' //)
C      END OF ENTRY CHSLVS (CHSLVX,CHSLVI).
       END

$
```


DAT
FILM